University of Rhode Island
DigitalCommons@URI

Open Access Master's Theses

2021

A NONLINEAR MODEL PREDICTIVE CONTROLLER FOR A WALL-ROLLING FULLY ACTUATED UAV

Matthew Little

Follow this and additional works at: https://digitalcommons.uri.edu/theses

A NONLINEAR MODEL PREDICTIVE CONTROLLER FOR A WALL-ROLLING FULLY ACTUATED UAV

BY

MATTHEW LITTLE

A THESIS SUBMITTED IN PARTIAL FULFILLMENT OF THE

REQUIREMENTS FOR THE DEGREE OF

MASTER OF SCIENCE

IN

ELECTRICAL ENGINEERING

UNIVERSITY OF RHODE ISLAND

2021

MASTER OF SCIENCE THESIS

OF

MATTHEW LITTLE

APPROVED:

Thesis Committee:

Major Professor Paolo Stegagno

Richard J. Vaccaro

Chengzhi Yuan

Brenton DeBoef
DEAN OF THE GRADUATE SCHOOL

UNIVERSITY OF RHODE ISLAND
2021

ABSTRACT

As Unmanned Aerial Vehicles (UAVs) become more commonplace in society they encounter greater risks due to crowded environments. In this thesis a novel solution for the locomotion of a Fully Actuated UAV is proposed, by having a UAV use a ring to roll along a surface. By rolling on the wall the UAV can use its point of contact to stabilize the UAV and anchor itself in the presence of wind disturbances and creating a more certain trajectory.

The Kinematic Rolling Model used to create the rolling motion uses 3 virtual Denavit-Hartenberg frames controlled by 3 parameters to define the position and orientation of the UAV from a given frame on a surface, with z being the normal. By creating a trajectory with these parameters the UAV's positional and rotational trajectories were calculated. After plugging the linear and angular accelerations into a Dynamic Model for a UAV, which was modified to include a normal force, the forces and required torques were found. This procedure was tested in both Matlab and ROS/Gazebo environments and found that the UAV was able to perform wall rolling. Though like other vehicles with nonholonomic constraints the UAV may be required to maneuver to reach a desired point.

To show possible methods of control of the UAV while wall rolling and in free flight, a Nonlinear Model Predictive Controller was developed with Feedback Linearization and Gradient Descent path planners. While each path planner was somewhat successful the Wall Rolling Feedback Linearization and Free Flight Gradient Descent path planners had drawbacks that hampered their usage.

Additionally the ability of the UAV to resist wind gust was simulated in the ROS/Gazebo environment. The UAV was found to be able to resist wind forces better than the UAV in free flight with comparable controllers. The ability to resist the wind was also dependent on the direction of the wind.

ACKNOWLEDGMENTS

I would like to thank my major professor Paolo Stegagno, who assisted me in navigating the thesis process, getting the needed software infrastructure, and sorting through the academic literature.

TABLE OF CONTENTS

LIST OF TABLES

TABLE PAGE

LIST OF FIGURES

FIGURE PAGE

CHAPTER 1

INTRODUCTION

While Unmanned Aerial Vehicles (UAVs) have become popular and useful in such areas as surveillance, recreation, and light shipping, they still have numerous faults. One of the largest issues is their inherent instability resulting from using thrust as the means of actuation. This forces UAVs to use a large amount of power in even the simplest maneuvers. If this is not done precisely a UAV can easily become unstable and cause a collision. Due to the large amount of power a UAV is using it can easily cause massive damage to its surroundings.

This is made significantly worse in the presence of wind or other disturbances. These disturbances can cause various problems ranging from increasing the amount of error in the UAV's assumed position, to forcing the UAV to maneuver to stay afloat, and could even cause the UAV to have a collision. In confined or congested areas these collisions are more likely due to the smaller space the UAV would have in which to maneuver. Studies have been done in hopes of better understanding and knowing how to counteract strong winds in free flight [1]. Some of the potential solutions have included using a proposed sliding mode controller to adapt to wind disturbances [2]. Other attempts have used an L1 Adaptive Velocity controller to counteract wind disturbances [16].

In this thesis we propose a different approach to this problem by turning the presence of obstacles such as walls into an opportunity. In fact, the proposed method for stabilizing the UAV will involve keeping it in contact with a wall during its motion. To do this a ring will be placed around the UAV that will remain in contact

with the wall. The UAV will stay in contact with the wall by rolling along the wall using the ring as a wheel. By having the UAV apply a normal force on the wall through the ring it can be used to anchor the UAV to the wall and allow it to better resist wind forces and improve overall stability. However, the proposed wall rolling motion causes a nonholonomic constraint into an otherwise holonomic, yet underactuated system. The combination of a nonholnomic constraint with the inherent underactuation of the UAV however, could result in unfeasible trajectories. Therefore, we have developed our controller for a fully actuated UAV (i.e., a UAV capable of independently maneuvering its position and orientation).

The developed controller needs to be able to move the UAV as part of the proposed wall rolling model as well as in free flight and be able to guide the transition between both models. To do this, a Nonlinear Model Predictive Controller (NMPC) will be used because the controller will be able to judge different proposed paths regardless of which model the UAV is using at any particular point. This requires that other controller or path planing methods be built and used to build paths that the NMPC can use. The paths generated by the path generators will need to be able to get the UAV to its target location, if possible, and maintain the stability of the UAV.

CHAPTER 2

REVIEW OF LITERATURE

This thesis explores using a modified fully actuated UAV to traverse a crowded environment. The UAV will also attempt to improve it navigation by using model predictive controllers. The relevant literature on related topics is discussed in this section.

2.1 Six Degree of Freedom UAVs

A six degree of freedom or fully actuated UAV has the ability to independently control the forces and torques exerted on the UAV's body in all axes. The UAV may not be able to fly in every orientation, but this mobility will allow it to accomplish difficult and delicate tasks more easily and with less risk. The design can also allow for easier control solutions even if the overall system is more complex.

There have been several designs that attempt to do this. The most common design that does this is a UAV with six arms with a rotor tilted in alternating directions at an angle [7,8,14]. The end result needs to be that by using different motor speeds, the applied force and torque of the UAV can be controlled in every direction independently. These drones can also be expanded to include more arms [9] and they will still be six degree of freedom drones as long as the resulting force and torques are all independent. There are also some other designs that attempt to move the rotors themselves as a way of controlling the force and torque applied on each axis [10].

2.2 Cages and Surface Contacts for UAV

Cages are often added to UAV's as a method of protection, especially in commercially available UAVs as DJI [12] or parrot [13]. Some cages have even been used to mitigate the effects of a collision by rolling around the UAV allowing them to remain in relative control [3].

In some other works, a contact with a wall is intentionally made in order to either perform the task or increase the UAV stability. In [4], the UAV was equipped with arms intended to stabilize the UAV. Additionally it also included the design with casters to allow the UAV to move across the surface while keeping the UAV stable. In [5], a wall sticking UAV used electro-magnetic hold mount elements to stick an ultrasound sensor probe on the surface of a refinery. In [6], an UAV was equipped with an arm designed to come in contact with a wall to apply a force on it.

Additionally some UAVs have used their cages as sensory devices in order to directly receive feedback from the environment [7].

2.3 Feedback Linearization Controller

Feedback Linearization is a technique in which a nonlinear system is transformed into a linear system (that is, nonlinear feedback is used to transform) and the dynamics of the nonlinear model are turned into a linear state space model. Feedback Linearization has been successfully applied to Six Degree of Freedom UAVs as a method to control them [8,14].

2.4 Nonlinear Model Predictive Controller

Model Predictive Controllers attempt to use a model of the system and test various possible paths in order to find the one that best fits the goals of the task [11]. This controller may also find the best solution even if it is not obvious. There are two main parts of this controller that are: a system to generate various possible paths or sets of inputs, and a cost function to determine which path meets the tasks needs best. This allows it to account for very complex systems that are not perfectly constrained. It can also be used to optimize variables that are not directly related to accomplishing the task of the system. Simulated UAVs have been successfully controlled with nonlinear model predictive controllers [9].

CHAPTER 3

MODELING

3.1 Kinematic Model

Since the UAV will be rolling along the wall it will be constrained in a manner similar to a wheel. This means that the point on the ring that is in contact with the wall cannot move but the entire UAV can rotate moving the forward or backwards by shifting the point of contact. Otherwise this point of contact can not move left or right or detach from the wall while using this model. In addition, the rotation of the UAV will need to be synced up with its velocity.

These limits constrain three dimensions, while the remaining three dimensions can be expressed as a sequence of three virtual Denavit-Hartenberg frames depicted in Figures 2, 3, and 4 with the entire chain being expressed in Figure 1 and summarized in Table 1. These can be summarized as the angle α of the instantaneous trajectory of the UAV on the wall, the angle θ that the segment AB connecting the point of contact between the UAV and the wall and the center of the UAV has with the wall plane, and an angle ϕ that controls how fast the UAV rolls. With this representation, the frames will be attached to a virtual mobile base moving on the wall with a linear velocity depending on the derivative of ϕ as expressed in equation 1.

$$v_0 = -y_1 r_{drone} \dot{\phi} \qquad 1$$

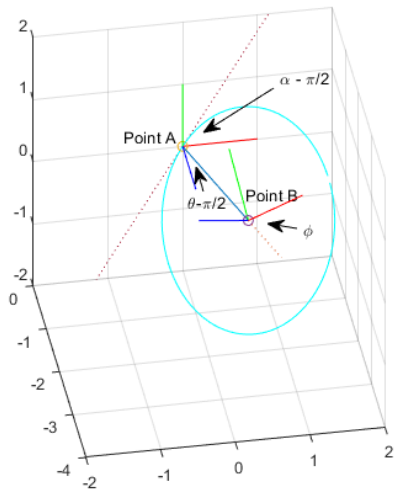

Figure 1: UAV Ring (cyan) with UAV wall trajectory(red) and Kinematic parameters

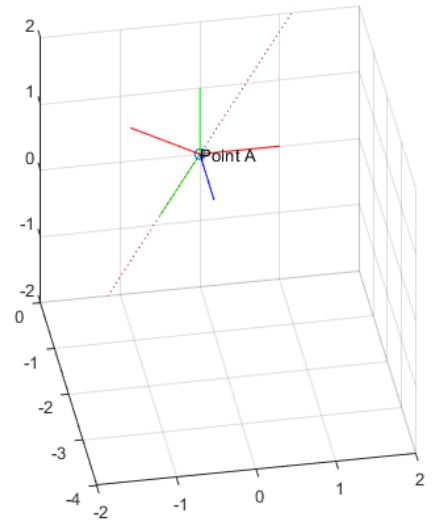

Figure 2: Transformation between Frame 0 (Base Frame) and Frame 1

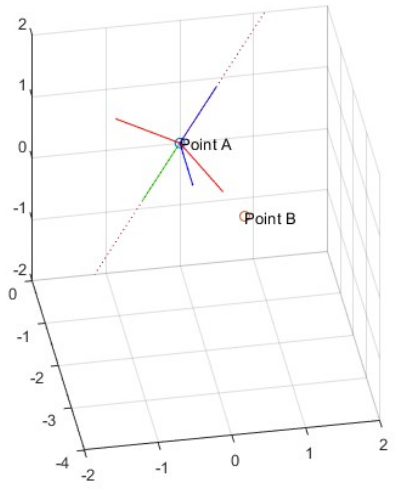

Figure 3: Transition between Frame 1 and Frame 2

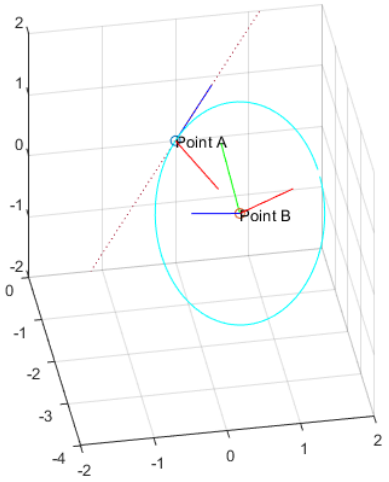

Figure 4: Transition between Frame 2 and Frame 3 (UAV's Body Frame)

Table 1: Virtual Denavit-Hartenberg parameters

Frame	d	θ	a_{i-1}	α_{i-1}
1	0	α	0	0
2	0	θ	0	90
3	0	φ	r_{drone}	90

3.2 Dynamic Models

Once the desired linear and angular acceleration are found by modeling the trajectory of the UAV from kinematic inputs a dynamic model of the UAV can be used to relate the linear and angular accelerations to the motor speeds. The dynamic model for a UAV in free flight is given in equations 2 to 4, however this model does not include the normal force that is presupposed in this stabilization method.

$$\ddot{p}=\begin{bmatrix}0\\0\\-g\end{bmatrix}+\frac{1}{m}R_B^W F u \quad (2)$$

$$\dot{\omega}_B=-I_B^{-1}(\omega_B\times I_B\omega_B)+I_B^{-1}Hu \quad (3)$$

$$\dot{R}_B^W=R_B^W[\omega_B]\wedge \quad (4)$$

Where p is the position of the UAV, g is the acceleration from gravity, m is its mass, R_B^W is the rotation matrix from the body to the world frame, F is the thrust coefficient matrix, u is the motor input vector, ω_B is the angular velocity of the body, I_B is the inertia of the body, H is the drag coefficient matrix, and $\wedge$ is the hat function.

The normal force is added to the dynamic model by adding a force and torque in a manner similar to the forces and torques from thrust. This adapted dynamic model can be seen in equations 5 to 7.

$$\ddot{p}=\begin{bmatrix}0\\0\\-g\end{bmatrix}+\frac{1}{m}R_B^W F u+\frac{1}{m}R_0^W\begin{bmatrix}0\\0\\N\end{bmatrix} \quad 5$$

$$\dot{\omega}_B=-I_B^{-1}(\omega_B\times I_B\omega_B)+I_B^{-1}Hu+I_B^{-1}(p_A^3\times R_0^3\begin{bmatrix}0\\0\\N\end{bmatrix}) \quad 6$$

$$\dot{R}_B^W=R_B^W[\omega_B]\wedge \quad 7$$

In these equations N is the magnitude of the normal force that the drone will produce, p_A^3 is the position from the body of the UAV to the contact point, A, which is also the location of frames 0, 1, and 2. R_0^3 is the rotation matrix from frame 0 to frame 3.

Both dynamic models can be resolved to a single solution only if the UAV has 6 independent rotors. If it has more it will need an additional way to resolve the equations to fit the desire accelerations.

3.3 Matlab Verification

The Kinematic and Dynamic Models were verified by generating and implementing the paths in a Matlab environment with a model UAV. The initial setup parameters of the Kinematic model were the point on a surface where the UAV would be in contact, 3 reasonable Kinematic inputs, and the relevant properties of the UAV such as mass, inertia, and the radius of the ring. By applying velocities and or accelerations of Kinematic parameters to the model, a path of desired positions and orientations were generated. These positions and orientations can be used to find the linear and angular acceleration required to accomplish the path derived from Kinematic Model. These accelerations, as well as further parameters from the UAV such as mass and inertia, can be used to solve the Dynamic Model, adjusted to include a set normal force, for the forces and torques the UAV is required to follow the

generated path. These forces and torques were reapplied to the standard Dynamic Model while applying any normal force that would be required to keep the ring of the UAV above the surface it was supposed to be rolling on.

The resulting simulations showed that the UAV responded correctly to any given inputs. Results from one of the simulation can be seen in Figure 5.

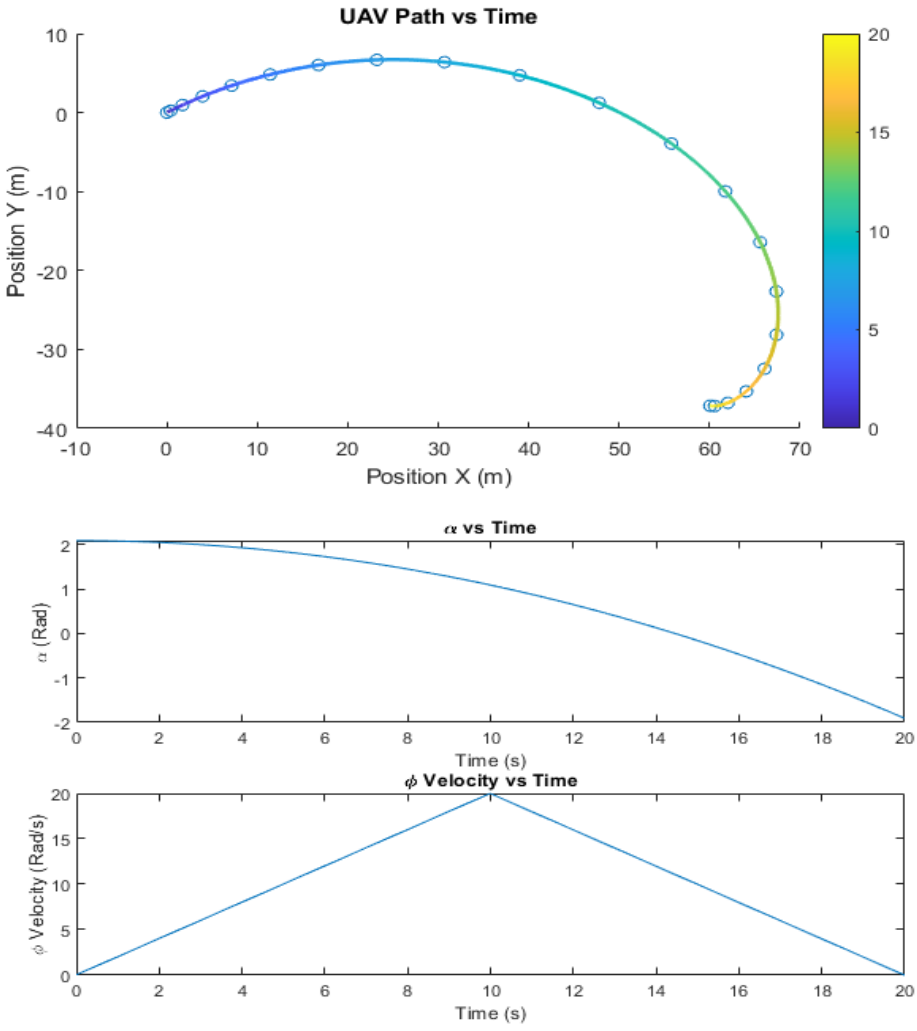

Figure 5: Matlab Verification Simulation Path

CHAPTER 4

GAZEBO SIMULATION SETUP AND ROLLING MOTION TESTS

Simulations of the fully-actuated UAV were performed in a ROS/Gazebo environment. All simulations were done on a Windows machine using the new Linux Command Line function. A graphics server was also used to allow the Linux graphics to be shown in Windows. The Gazebo simulation was controlled with inputs from the "rqt_gui" interface, which is part of ROS. In addition to providing inputs the "rqt_gui" interface also recorded the data in bag files. All coding and file modifications were done in VSCode. All compilation was done using the Catkin workspace system. The graphical elements of this setup can be seen in Figure 6.

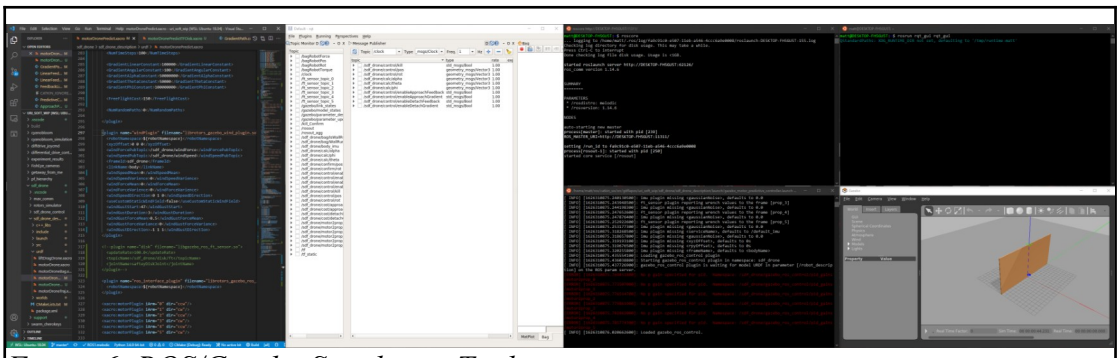

Figure 6: ROS/Gazebo Simulation Tools

4.1 Building the UAV

After the basic environment was set the full actuated UAV was built using the native ROS file system. This UAV achieved its maneuverability by having six rotors with tilted propellers as first presented in [7,8,14]. The basic motor operation was implemented through native components of the ROS system. However, getting the UAV to fly required additional plugins to calculate and create the thrust and drag the propellers would apply to the UAV while spinning at certain speeds. The

"rotors_gazebo_plugins" contains a plugin that was selected for this purpose. A disk was also added to the UAV to allow it place of a ring or cage to allow the UAV to come in contact with the wall and roll. The graphical model of the UAV can be see in Figure 7. An IMU was initially considered but was replaced with direct model measurements that would be 100% accurate. All UAV controllers are implemented as single plugins.

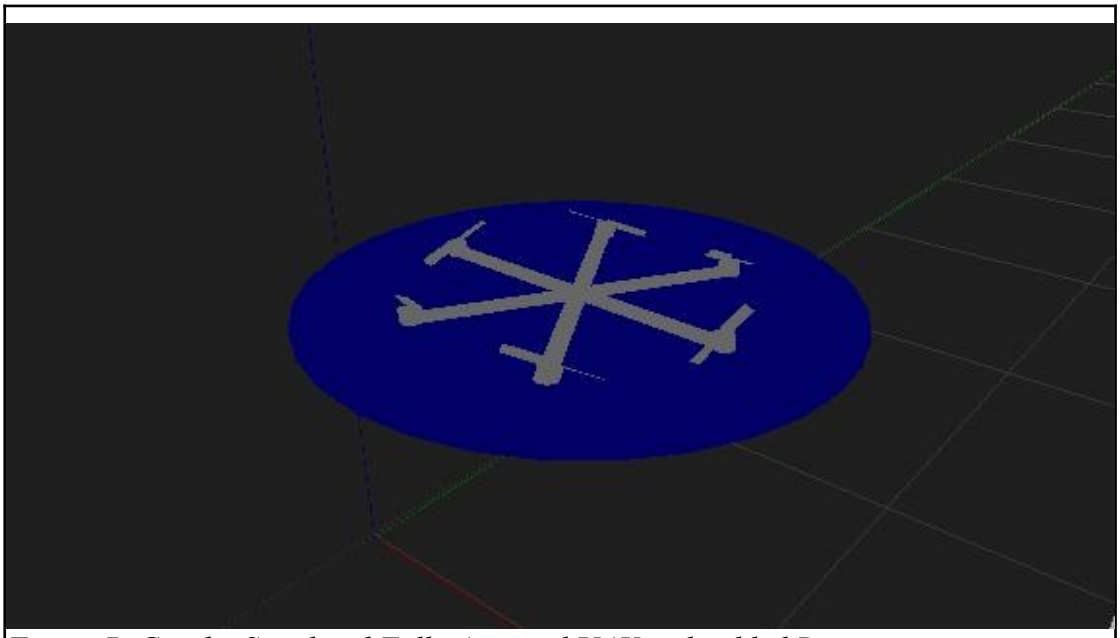

Figure 7: Gazebo Simulated Fully Actuated UAV with added Ring

4.2 Feedback Linearization Controller

A Feedback Linearization controller presented in [8,14] was added to ensure that UAV could be controlled through the motor inputs. This controller achieved the simple task of stabilizing and controlling the position and orientation of the UAV in free flight. The UAV finds the error between the robots current position and orientation and its desired position and orientation, as well as the derivative and integral for each. The error of the position is simply the difference of the two points and can be seen in equations 9,

$$e_p = p - p_d \tag{9}$$

where e_p is the error in position and p_d is the desired position. The orientation used for this controller is based on the 3D rotation group or SO(3). The error between the current and desired orientations can be seen in equation 10 and defined in [7,8,14], where e_R is the error in the rotation, R_d^W is the desired rotation and $\vee$ is the vee function.

$$e_R = \frac{1}{2}\left(\left(R_d^W\right)^T R_B^W - \left(R_B^W\right)^T R_d^W\right)^\vee \tag{10}$$

The error between the current and desired angular velocity can be seen in equation 11,

$$e_\omega = \omega_B - \left(R_B^W\right)^T R_d^W \omega_d \tag{11}$$

where e_ω is the error in angular velocity and ω_d is the desired angular velocity. It takes this difference and uses PID controllers to find a linear and angular acceleration, and therefore forces and torques, that would move the UAV to its desired state. These can be seen in equations 12 and 13,

$$\ddot{p} = \ddot{p}_d - K_{p1}\dot{e}_p - K_{p2}e_p - K_{p3}\int e_p \tag{12}$$

$$\dot{\omega}_B = \dot{\omega}_d - K_{R1}\dot{e}_\omega - K_{R2}e_R - K_{R3}\int e_R \quad (13)$$

where all K values are constants used in the PID controllers. The robot then uses a Jacobian, or rather the inverse of the Jacobian, derived from the dynamic equations listed as equations 2 and 3, to translate the required forces and torques to motor speeds, which are broadcast to the native ROS motor controllers.

The Feedback Linearization controller was used as a test case to prove that the Gazebo simulation, ROS environment, and all the plugins, could all successfully communicate with the controller plugin. It also showed that many of the common functions that would be later used in other controllers were correct.

The environment used to test this controller was an empty world. The tests conducted were having the UAV do some simple maneuvers by assuming different positions and orientations. In Figures 8 and 9 we report the actual and desired position of the UAV with respect to time in a positioning experiment. The plots show that the UAV was able to achieve the desired position and orientation.

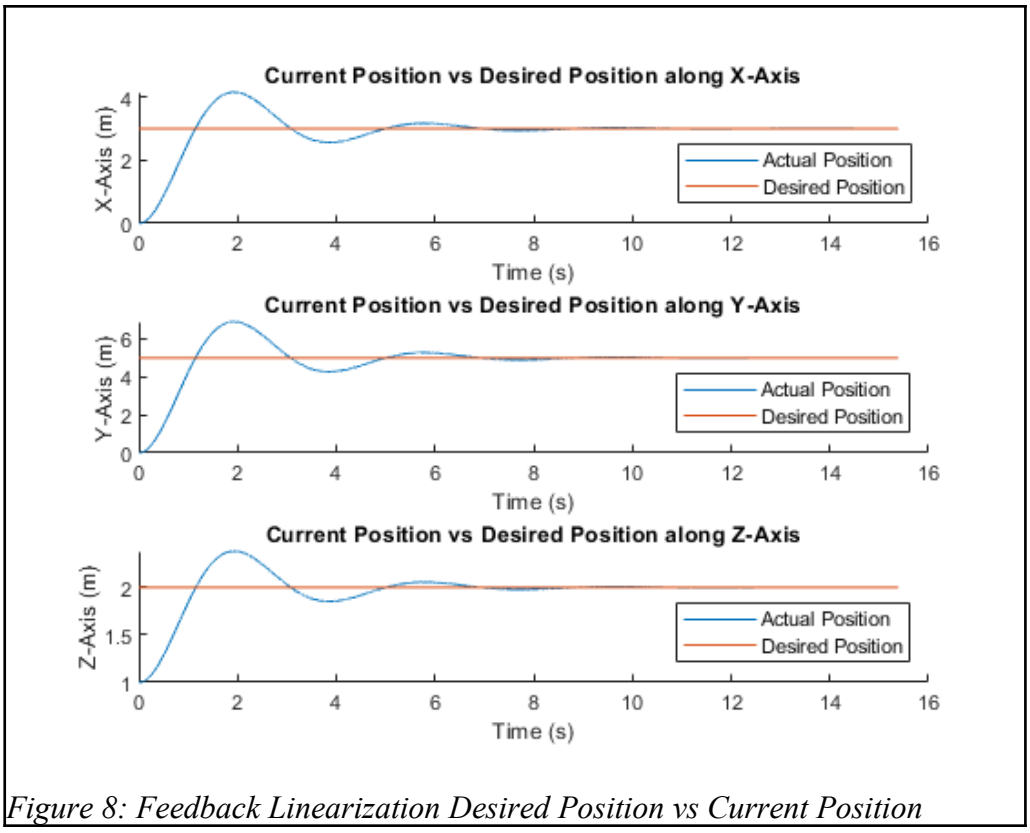

Figure 8: Feedback Linearization Desired Position vs Current Position

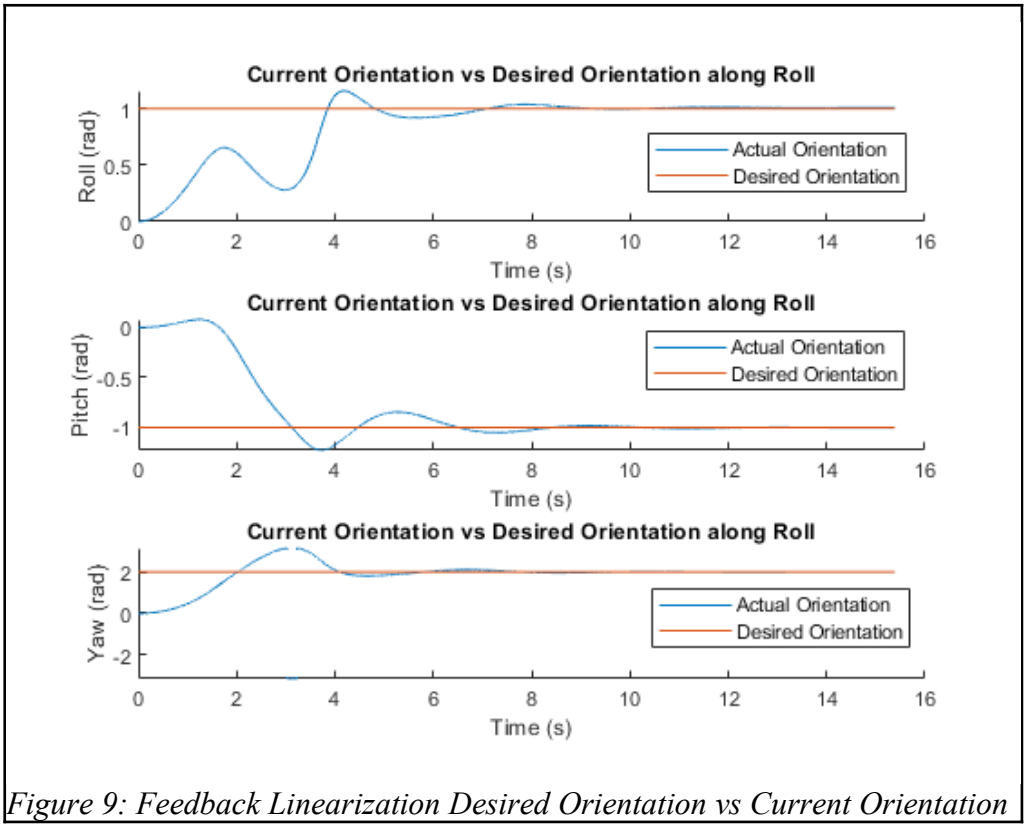

Figure 9: Feedback Linearization Desired Orientation vs Current Orientation

4.3 Open Loop Rolling Motion Trajectories

To test the Kinematic Rolling Model (KRM), an open loop controller was built. The controller will run a simulation of the KRM and will calculate the motor speeds required to achieve the new position in the next time step. This controller therefore cannot jump from one Kinematic configuration to another very different configuration but this controller will allow a trajectory of close-together configurations to be smoothly turned into a trajectory in space. The Kinematic trajectory will also be created in this controller by using preset initial values and receiving broadcasted double derivatives of the Kinematic parameters α, θ, and ϕ sent by the "rqt_gui" interface.

Once the controllers receive the broadcasted Kinematic accelerations they will be used to continuously recalculate the velocities as well as the Kinematic parameters themselves. By inputting the Kinematic parameters and $\dot{\phi}$ at each point in time into the Kinematic Rolling Model the position and orientation of the UAV can be found at each corresponding point. The linear acceleration needed to perform the maneuver can be found by numerically finding the double derivative of the position. The angular acceleration, which is the derivative of the angular velocity, is calculated through equation 7 using the UAVs orientation as a rotation matrix and the derivative of the rotation matrix. The angular and linear accelerations are then plugged into the Modified Dynamic Model in equations 5 and 6 to find the forces and torques the UAV will need to apply. The Modified Dynamic Model includes a normal force that the UAV will be receiving from the wall onto its ring and is set as part of the UAV controller plugin. The forces and torques are then converted using the inverse of a

Jacobian matrix into the squared motor speeds and, by extension, the motor speeds. The motor speeds are broadcast to their corresponding motor controllers.

This controller will be tested in a world with a single wall for it to press against. The controller correctly assumes that the UAV starts from a set of the Kinematic parameters that will make the UAV level, one meter off the ground, and touching the wall as can be seen in Figure 10.

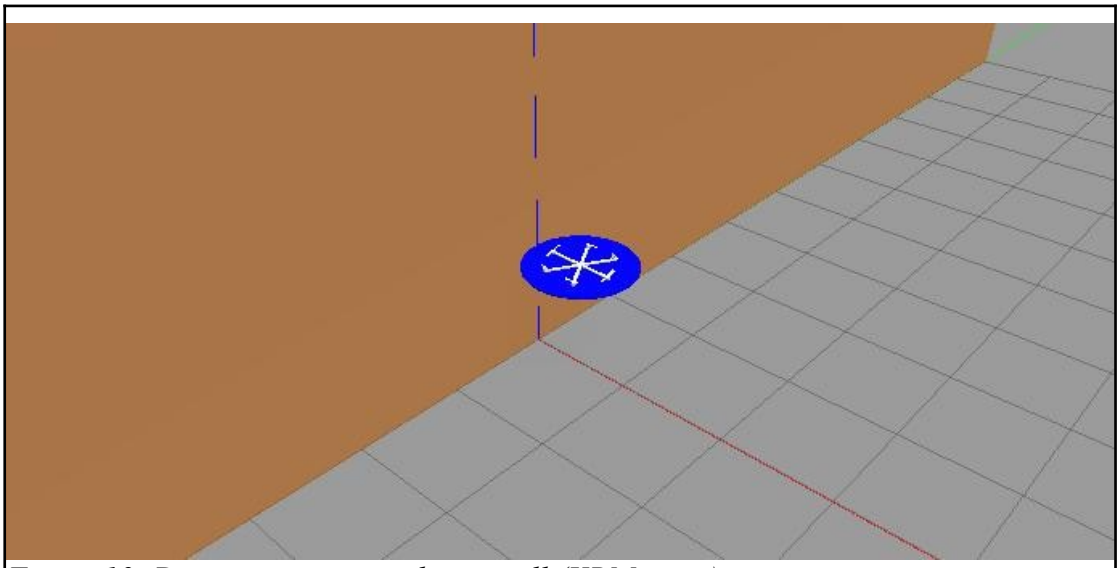

Figure 10: Drone in contact with test wall (KRM setup)

The tests of the KRM open loop controller successfully demonstrated that the KRM model performs as expected. In Figures 11 - 13 the open loop controller was compared to the simulation of the same Kinematic trajectories to their intended output from the Matlab verification of the KRM. In most simulations where the drone moved it did destabilize and became uncontrollable. This is assumed to occur due to the instability inherent in UAVs, as it itself is not a stable platform.

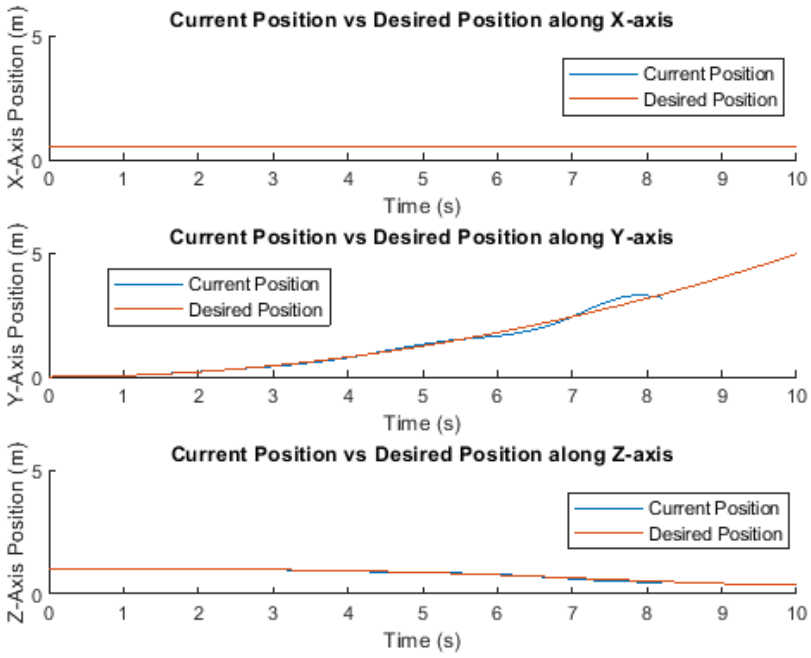

Figure 11: Position of the KRM open loop simulation and validation data

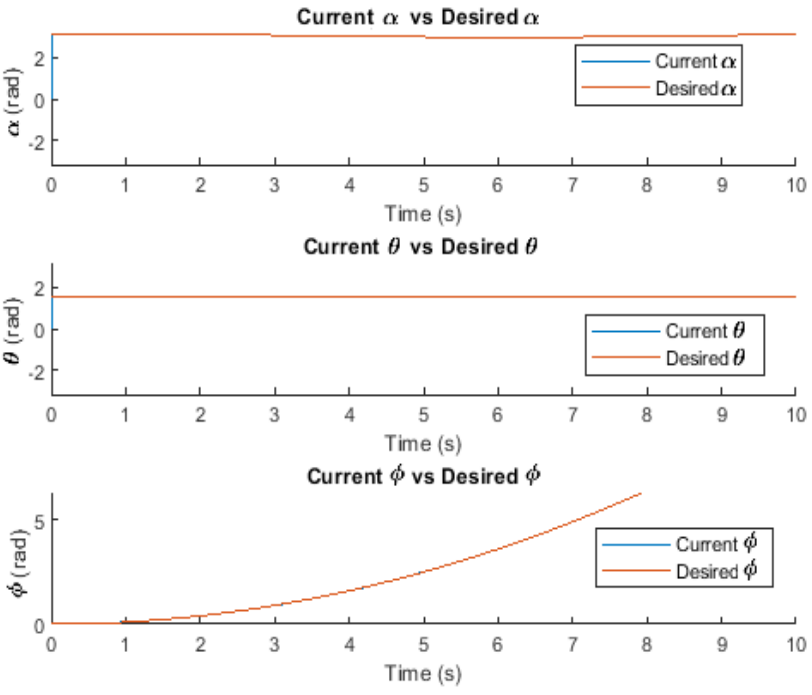

Figure 12: Orientation of the KRM open loop simulation and validation data

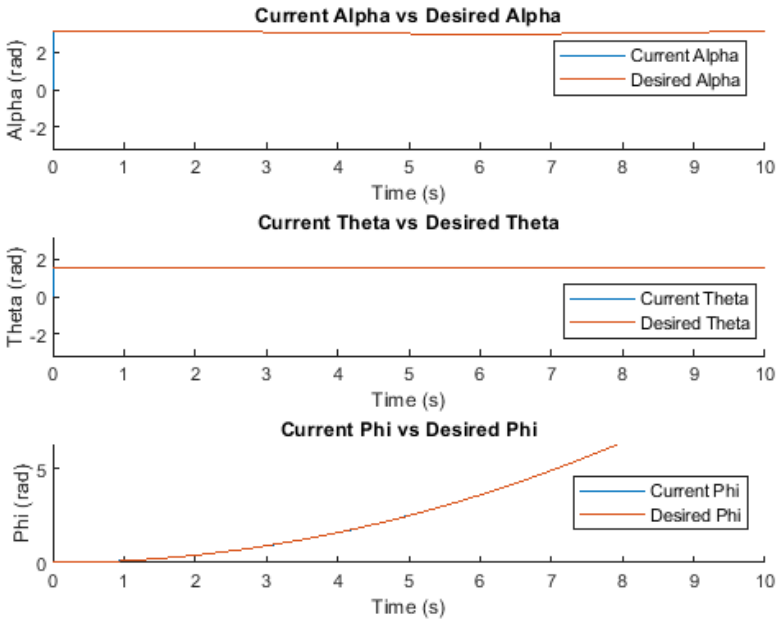

Figure 13: Kinematic parameters of the KRM open loop simulation and validation data

CHAPTER 5

NONLINEAR MODEL PREDICTIVE CONTROLLER

An additional higher-level loop of control has been added to act as path planner with continuous replanning. In this loop, the position and orientation of the UAV will be controlled through a Nonlinear Model Predictive Controller (NMPC). This control paradigm was selected because the UAV is modeled as a switching system, with different models depending on it current location and orientation as well as those of the prospective paths. Specifically, the switching condition depends on whether or not the UAV is or will be in contact with the wall. Since the NMPC can map out paths on both the wall and in free flight it can determine which is best according to a cost function. To make the NMPC useful it needs several path generators to give it a wide array of different paths to choose from. The best path is then chosen by comparing the different paths using a cost function. A flowchart of the NMPC can be seen in Figure 14.

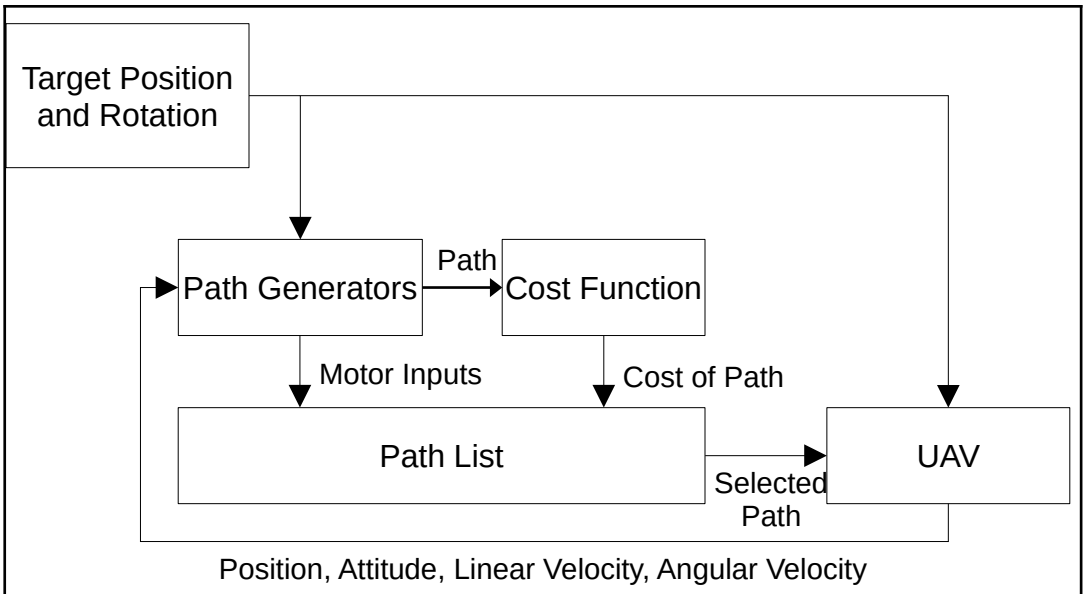

Figure 14: Nonlinear Model Predictive Controller Flowchart

5.1 Path Generation

The NMPC requires an assortment of paths to be able to operate effectively. The variety of these paths will also improve the NMPC by giving more meaningful paths to choose from as opposed to similar paths which give very little difference to choose from. To this end several path generators were created that attempt to move the UAV closer to the target using different paths.

The first form of path generation was to use random numbers as inputs to whichever model the UAV would be in. This process of using a random input with NMPC was able to move the UAV to the target. The method was highly computationally expensive and was slow or unstable depending on the constant used to bound it. This method was not used in favor of more optimized methods.

The second path generator designed for use by the NMPC uses a Gradient Descent method [15] to create a path. This involves using the gradient of the cost function of the NMPC (introduced in section 5.2) with respect to each input parameter

at the current point, to move the UAV toward the desired end state. A constant was also used to take the gradients and turn them into prospective inputs. By simulating the path created for the UAV with these inputs a path is made for the NMPC. Of the two ways to find the gradient, analytically or numerically, it was easier to find the gradient numerically since it allowed the controller simulation to exist as the cost function changed around it without needing to worry about recalculating the derivative equation. The controller will then use the gradient to make the input parameters. Since this method of finding the gradient relies on interpolation it may contain some error. The Gradient Descent method was able to generate useful paths in both free flight and wall running conditions.

The third path generator used in the NMPC was the Feedback Linearization Controller described previously and adapted to fit in the NMPC. This includes using a PD controller instead of a PID controller in the feedback mechanism (equations 12 and 13). Since the controller for the NMPC can switch between models, trying to keep the integral term could cause problem for the controller due to the presence of the wall that would force the UAV to a constant error in one direction, causing the integral term to increase indiscriminately.

Another Feedback Linearization method was used to create a path by changing the α parameter so the drone's direction of movement is in line with the target point. It also changes the ϕ parameter to move the UAV in the direction of the point. The inputted Kinematic acceleration parameters were controlled through PD controllers as can be seen in equations 14-17.

$$\ddot{\alpha} = k_{\alpha 1}(\dot{\alpha}_d - \dot{\alpha}) + k_{\alpha 2}(\alpha_d - \alpha) \qquad 14$$

$$\alpha_d = atan2(y^0_{a_d}, x^0_{a_d}) + \frac{\pi}{2} \quad 15$$

$$\ddot{\theta} = k_{\theta 1}(\dot{\theta}_d - \dot{\theta}) + k_{\theta 2}(\theta_d - \theta) \quad 16$$

$$\ddot{\phi} = k_{\phi 1}(\dot{\phi}_d - \dot{\phi}) + k_{\phi 2}(|a^0_d|) \quad , \quad 17$$

where all k values are constant gains used in the PD controllers, a^0_d is the vector from the UAV's current point of contact to the target point of contact or the point on the wall closest to the desired point. The variables $x^0_{a_d}$ and $y^0_{a_d}$ are the x and y components of a^0_d respectively. α_d is the desired α parameter that will steer the UAV to the target point using a positive $\dot{\phi}$. Since the UAV may move in either direction perpendicular to the UAV's z vector, extra logic was added to allow the controller to choose if it should realign with α_d or $\alpha_d + \pi$ based on which angle is smaller, and drive the ϕ parameter in a corresponding manner. The θ_d variable is also kept set at $\pi/2$ to keep the UAV as stable as possible. All KRM parameters are changed using PD controllers as well.

Finally, to make sure that the prospect of attaching to the wall was a possibility, methods were also added to make paths that would bring the UAV in contact with the wall as well as detach from it. This is important since most paths the UAV might immediately consider would not be moving the UAV to the wall. If no path moved the UAV to the wall then it has no chance to see if moving to and resting on the wall will be beneficial in the long term. In addition, none of the paths using the KRM would attempt to detach from the wall and specific methods would need to be activated in order to have the UAV attempt to detach.

5.2 Cost Function Selection

The cost function of the controller is what is used to determine which path proposed by the Path Generators leaves the UAV in the most desirable state. The cost function can be designed so that it will be able to take different parameters into account depending on what model is being used. This allows the cost function to objectively account for which path costs the least according to its parameters. Additionally the cost function can be changed depending on what model it is currently in use. The cost formula can bee seen in equation 19. The description of each parameter and when they are applied can be seen below:

$$cost = \sum_{timeStep=1}^{totalSteps} \left(k_1|e_p|^{k_7} + k_2|e_v|^{k_7} + k_3|e_R|^{k_8} + k_4|e_\omega|^{k_8} + k_5|e_\alpha|^{k_9} + k_5|e_{\dot{\alpha}}|^{k_9} + k_6|e_\theta|^{k_{10}} + \sin(\beta)\, pen_{ff}\right) \quad 19$$

- Position Error (e_p) – The absolute value of the positional error is used to move the UAV closer to the desired position.
- Velocity Error (e_v) - The absolute value of the velocity error is used to slow down the UAV.
- Rotational Error (e_R) (Free Flight only) – The absolute value of the rotational error, calculated using SO(3) parameters which can be seen in equation 10, is added to move the UAV to a desired orientation.
- Angular Velocity Error (e_ω) (Free Flight only) – The absolute value of the angular velocity error is used to slow down the rotation of the UAV.
- α Error (e_α) (Wall Rolling only) – The absolute value of the difference between α and the set desired state of $\pi/2$ attempts to keep the UAV level. The UAV should still be able to move diagonally if the cost gained by this factor is

outweighed by a decrease due to a change in position, allowing the UAV to move diagonally to an extent.

- $\dot{\alpha}$ Error ($e_{\dot{\alpha}}$) (Wall Rolling only) – The absolute value of the difference between the current change in α over time and the set desired state of 0 is used to slow down the change in α to stabilize it.
- θ Error (e_{θ}) (Wall Rolling only) – The absolute value of the difference between θ and the set desired state of $\pi/2$ will drive the UAV to stay perpendicular to the wall.
- Free Flight Penalty – ($sin(\beta)pen_{ff}$) (Free Flight only)- Where pen_{ff} is the free flight penalty value and β is the angle from the normal of the wall to a vector from the nearest point on the wall to the UAV. A penalty was added to free flight so the UAV would be able to weight the safety of staying rooted by the wall. Additionally a sin function was added to remove the penalty if the UAV was as to the target point as the wall would allow.
- k-values – all k values are constant gains.

The cost function will also have a second use when it is used to build the path of the Gradient Descent path generators. As such, some small changes were made to alter the behavior of those path generators. For instance, adding cost for the positional velocity and angular velocity was required to control the UAV with the Free Flight Gradient Descent path. The velocity of the KRM α and ϕ parameters was also required to control the Wall Rolling Gradient Descent path. However, since the positional velocity was also ideally related to the $\dot{\phi}$ it was used in place of the $\dot{\phi}$.

CHAPTER 6

NMPC SIMULATION RESULTS

The simulation results first needed to show if and how each component method works and see them all work together. The wall rolling components will need to be able to move the UAV to the point on the wall closest to the desired point in space. The free flight component should move the UAV to the desired position and orientation. Both of these will be necessary to see if the NMPC can be used to decide to both approach the wall, assuming it will be safer, and detach to reach the desired point in a single run. Finally the premise that using the wall as an anchor does improve stability needs to be tested.

6.1 Closed Loop Rolling Motion Trajectories

To be able to utilize the Kinematic Rolling Model (KRM) to navigate the wall a closed loop controller will be needed to guide the UAV to a target point. To create a closed loop controller the Gradient Descent controller and the Linear Feedback controller were applied to the UAV by isolating all but those paths in the NMPC. The orientation of the UAV was used to calculate the Kinematic, α, θ, and ϕ, parameters and provide the feedback to the controller. The equations for α and θ can be seen in equations 21 and 22.

$$path = Z_B^W \times N_{wall}^W \tag{20}$$

$$\alpha = atan2\left(\frac{y_{path}^0}{x_{path}^0}\right) + \frac{\pi}{2} \tag{21}$$

$$\theta = acos\left(-Z_B^W \cdot N_{wall}^W\right) \tag{22}$$

Where $path$ is the vector for the forward direction of travel on the wall, Z_B^W is the z-vector of the UAV body frame, N_{wall}^W is the normal vector of the wall, y_{path}^0 and x_{path}^0 are the path vector's y and x components respectively in the wall frame. The ϕ parameter was derived by calculating the rotation of the UAV with the previously derived α and θ parameters but with 0 for ϕ and finding the angle from the calculated frame to the UAV body frame about the shared z-axis.

Several simulations were done to see if the UAV was able to successfully reach different points traveling on the wall with the Gradient Descent method. In many of the initial tests the Gradient Descent path generator had difficulty choosing α acceleration. It became apparent that the UAV needed a significant linear velocity to make any change in α that would be based on a change in the position. An optimization based on this observation was done by calculating the values for interpolation for α to give $\dot{\phi}$ a constant value based on its current direction of travel. This gives a meaningful difference to the equation required to calculate an α gradient.

The Gradient Descent controller, while improved, still had difficulty reaching the target point but was able to approach it as can be seen in Figure 15. This was predicted to happen because of the cost incurred by having a cost associated with an e_α parameter, where the α parameter would incur an increasing cost the farther it was from $\pi/2$. However if the α parameter as well as the $\dot{\alpha}$ parameter were removed from the cost function it would cause instability and the UAV would eventually crash.

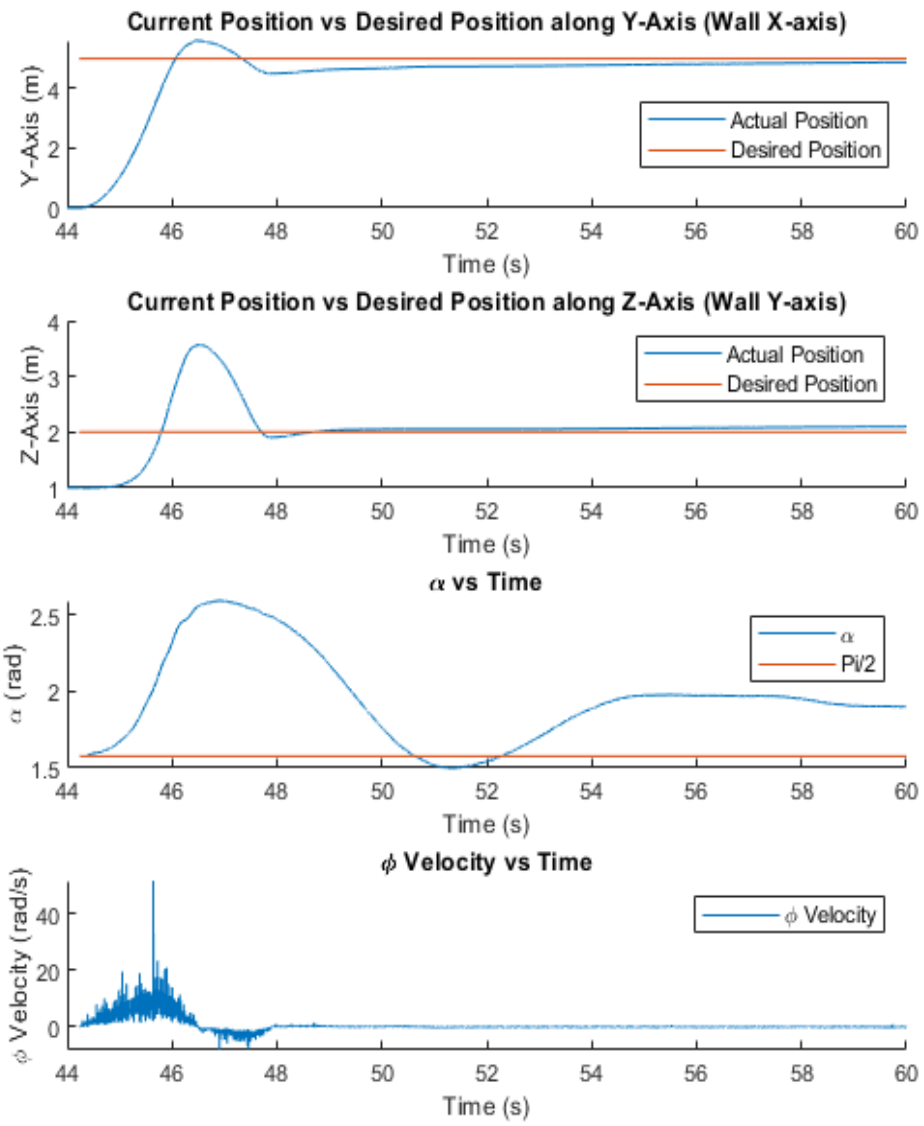

Figure 15: Gradient Descent Wall Rolling: Wall Position, α, and $\dot{\phi}$

The Feedback Linearization Wall Rolling controller was able move the UAV to the desired point on the wall. However, since the controller does not take into account any of the stability measures in the cost function it may attempt to climb right up the wall. Additionally when the UAV is close to the point on the wall near the target position it would spin rapidly in an attempt to realign and move the UAV at the same

time. The results of the simulations with the Linear Feedback Wall Rolling controller can be seen in Figure 16.

When the two wall rolling methods were combined within the Nonlinear Model Predictive Controller they did not perform as well as their two component wall rolling methods. Attempts to combine the two rolling methods either did not move the UAV close enough to be more effective then the Gradient Descent method alone or could not approach the point without assuming an unstable position. Attempts including shortening length of paths and altering the parameter that would influence each path individually did give some marginal improvement to the individual results, but did not improve the results significantly enough so it was later discarded in further testing.

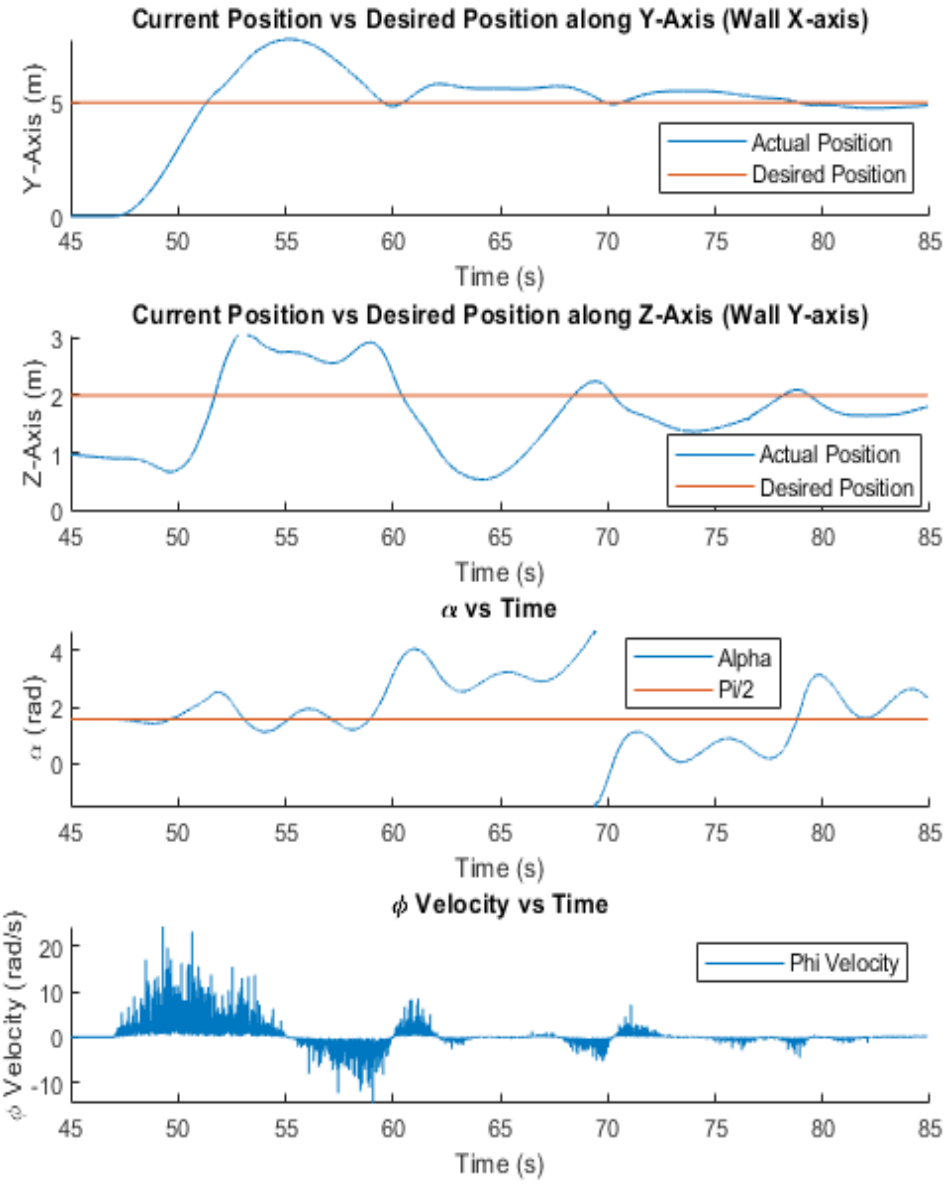

Figure 16: Feedback Linearization Wall Rolling: Wall Position, α, and $\dot{\phi}$

6.2 Closed Loop Free Flight Trajectories

The Gradient Descent controller was able to reach the desired point in space. It contained two major drawbacks. First, it was able to hold the UAV stable and make small changes to orientation, as can be seen in Figure 17 and 21.

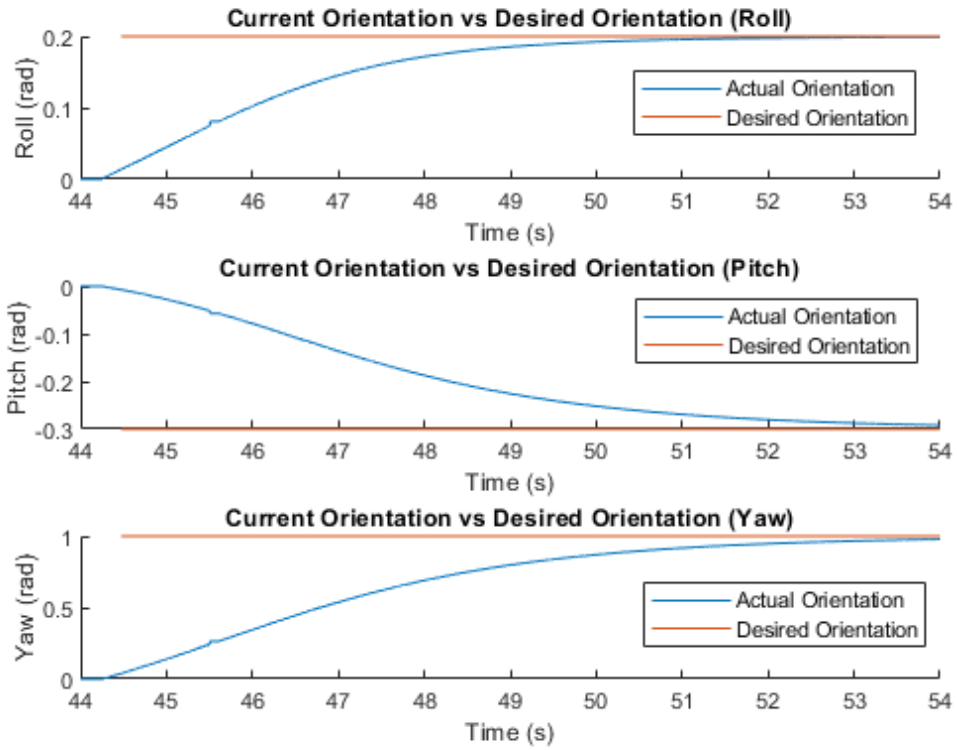

Figure 17: Gradient Descent Method: Small Change in Orientation

However, larger more complex changes get stuck as can be seen in Figure 18. Using small changes a path can be made to eventually move the UAV to the final desired position as can be seen in Figure 19. The Gradient Descent controller was able to move the UAV to the desired position, but due to changes in the cost function, it became less viable as the increase in speed will no longer be able to be slowed until it passes the desired position as can be seen in Figure 20.

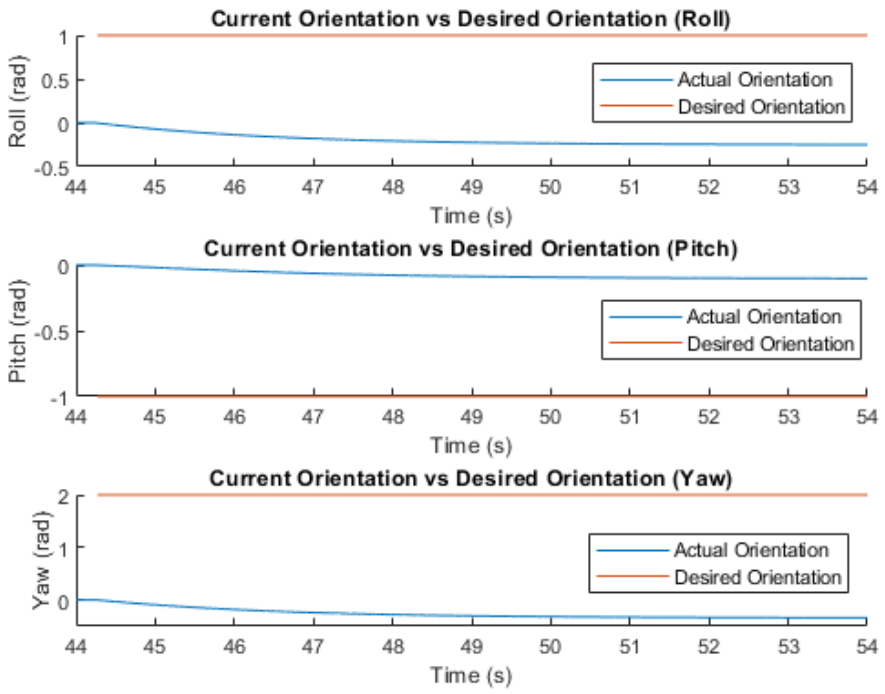

Figure 18: Gradient Descent Method: Failure to make complex change in orientation

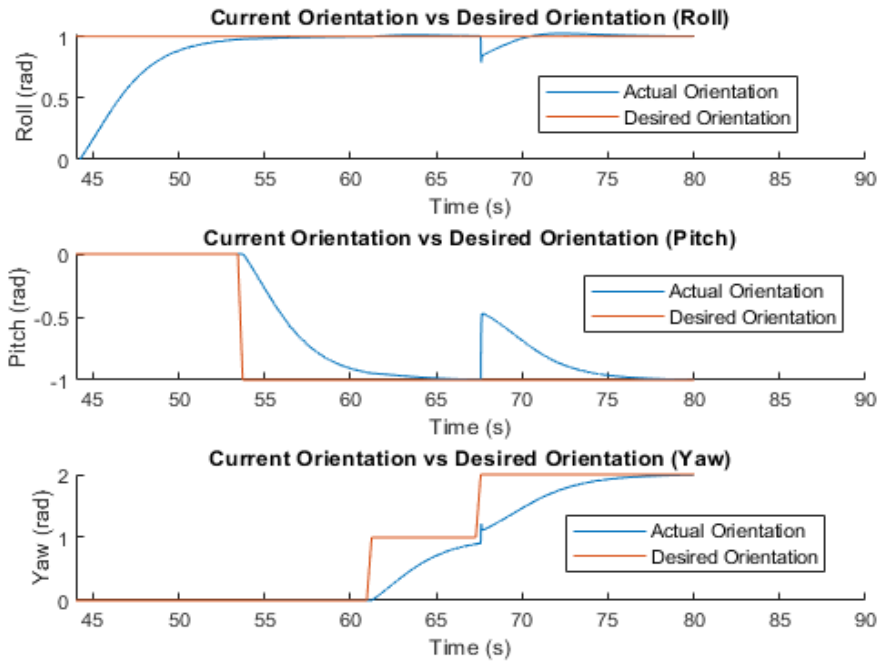

Figure 19: Gradient Descent Method: Using steps to move UAV to desired orientation

The Feedback Linearization controller performed as it did previously and was able to move the UAV to the desired position and orientation as expected, as can be seen in Figures 20 and 21. It was also found, however, that if the free flight controllers were combined in the NMPC that it performed better in moving the UAV to a position and performed just as well move to an orientation as can be seen in Figures 20 and 21 respectively.

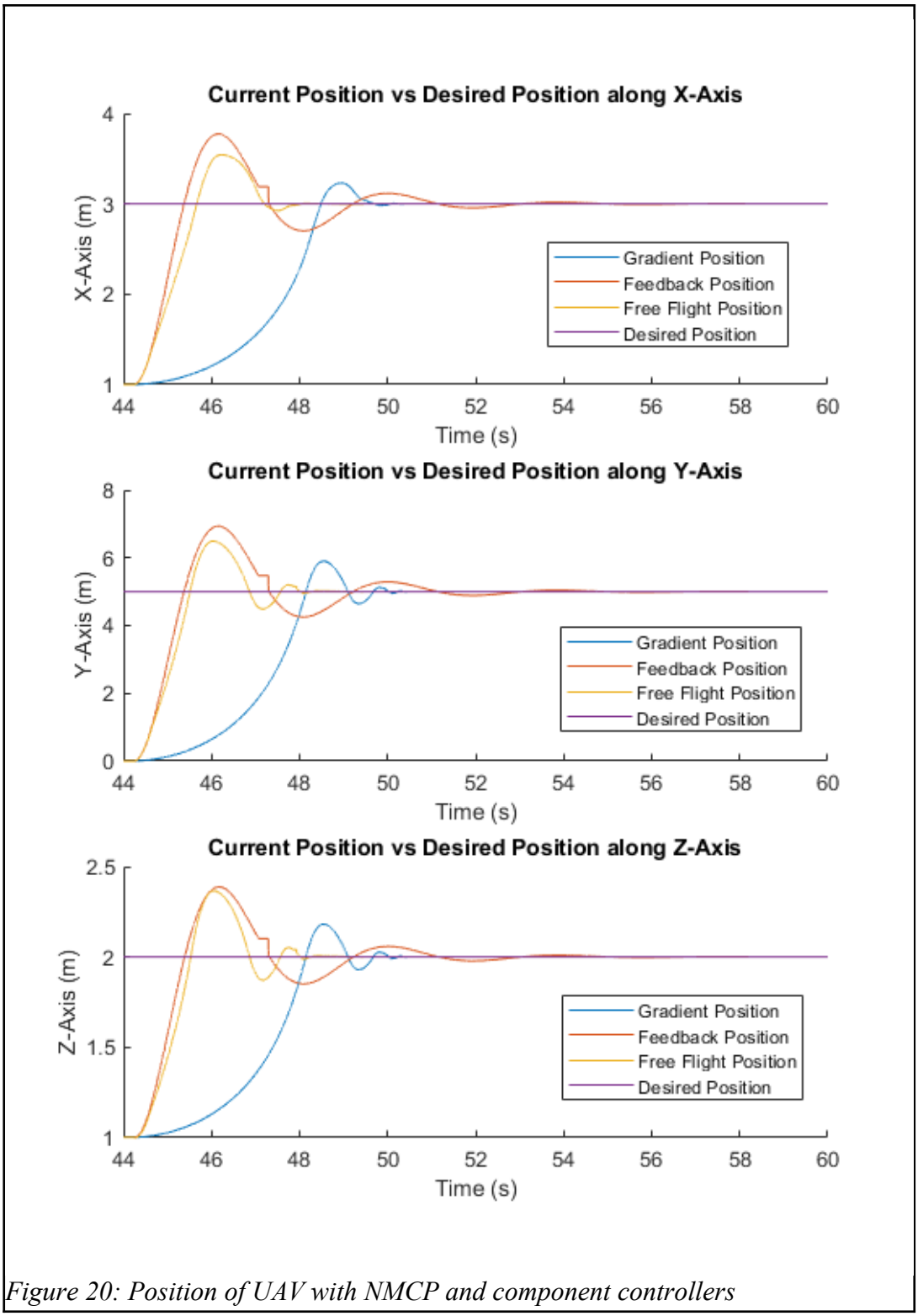

Figure 20: Position of UAV with NMCP and component controllers

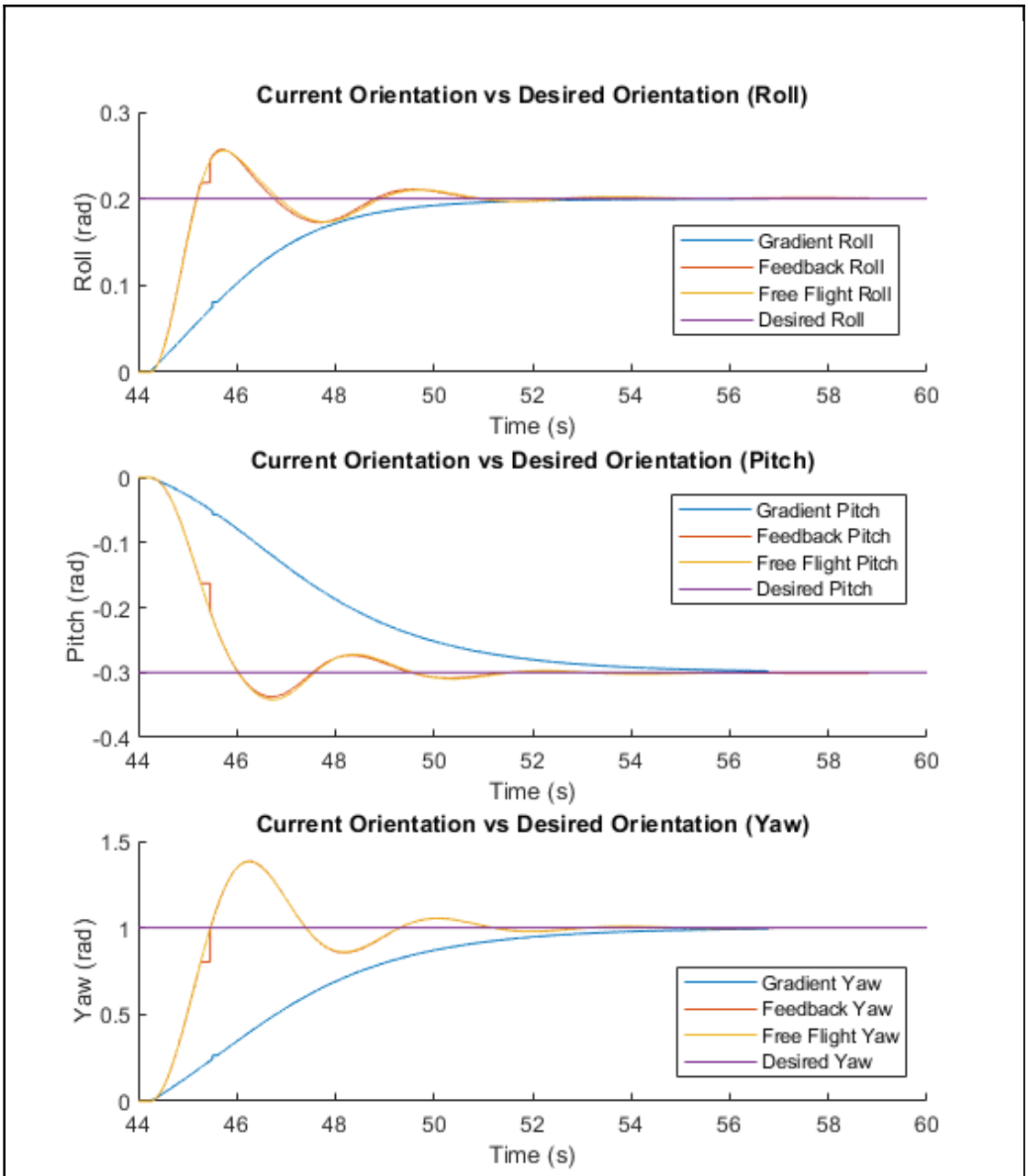

Figure 21: Orientation of UAV with NMPC and component controllers

6.3 Nonlinear Model Predictive Controller

The final Nonlinear Model Predictive Controller attempted to integrate the Kinematic Rolling Model with the normal Free Flight Model. Specifically this

combines the Gradient Wall Rolling Controller with Feedback Linearization Free Flight controller. The NMPC was able to move the UAV to a target point. After making the assumption that there is a benefit to using the wall to stabilize the UAV, a penalty was added to remaining in free flight to the cost function. If the NMPC finds that it costs less to both approach the wall and roll along it than to move directly to the target, then it will choose the former. The same should occur with the UAV detaching from the wall when the benefits of moving away from the wall outweighs the penalty of free flight. Both these should also be able to be done together as can be seen in figures 22, 23, and 24.

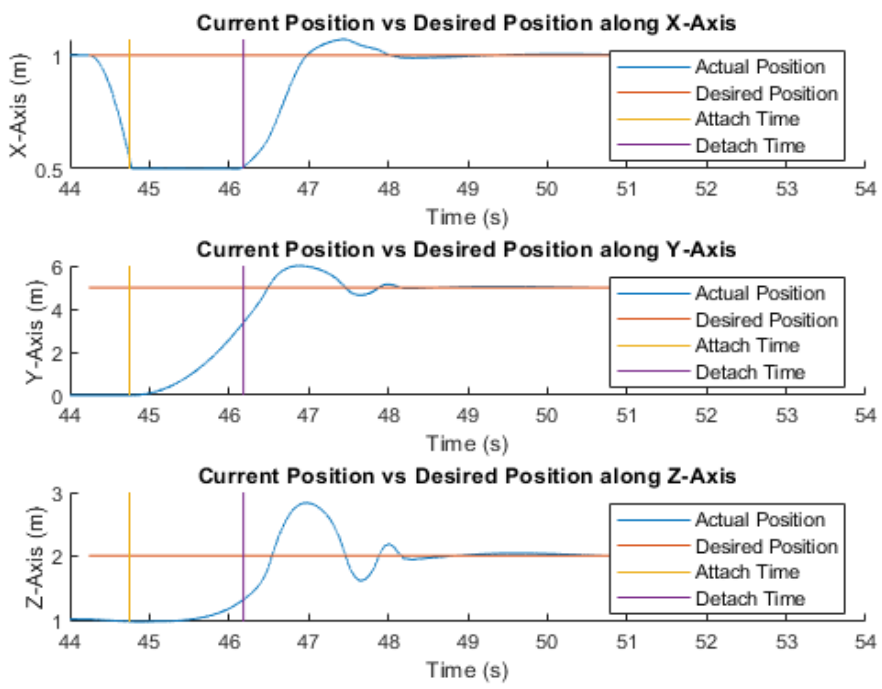

Figure 22: NMPC Combined Wall Running and Free Flight Positions

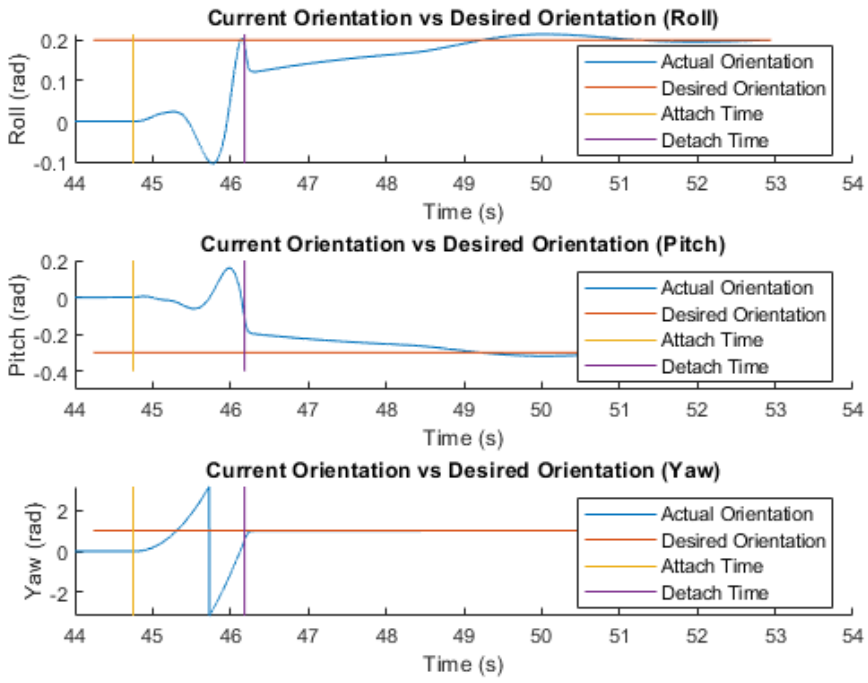

Figure 23: NMPC Combined Wall Running and Free Flight Orientations

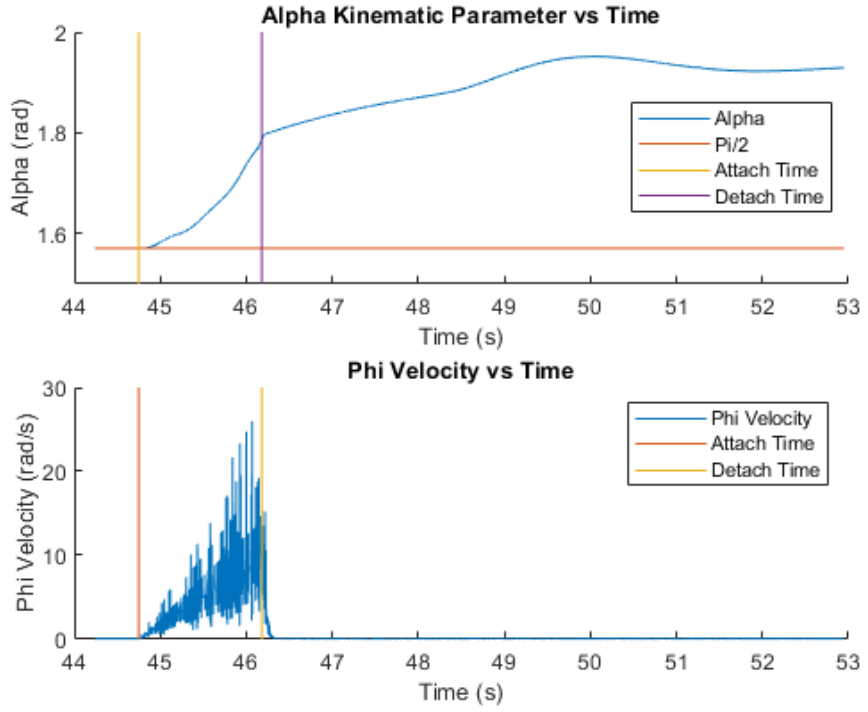

Figure 24: NMPC Combined Wall Running and Free Flight Wall Rolling Parameters

6.4 Stabilization of the UAV during Wind Gusts

One of the suspected benefits of using a wall rolling method was to help in the stabilization of the UAV by anchoring itself to the wall through its point of contact. This allows it to use the friction between the wall and the UAV to slow the UAV in motion parallel to the wall. It should also not allow the UAV to gain velocity and momentum in the direction perpendicular to the wall.

For these tests the UAV had a mass of 0.049kg and attempted to exert a force of 1N on the wall in an attempt to receive a normal force of the same magnitude. Unless otherwise stated the UAV will be attempting to remain in the same position on the wall. The “rotors_gazebo_wind_plugin” from the “rotors_gazebo_plugins” was then used to apply wind by applying a force for a set duration, which in all test was 3s, and from a set direction.

The simulation showed that the UAV was able to transfer the force from the wind into the wall as can be seen in by UAV not moving in figure 25, where very strong wind is applied in the -x direction directly into the wall. While wind parallel to the direction of travel of the UAV was able to move the UAV it was able to compensate and resist as well as reset its position after the wind stopped. This can be seen in figure 26 where wind was applied in the +y direction. Wind perpendicular to the original direction of travel and the normal force caused the UAV to shake along θ parameter and, depending on how strong, turned the UAV, along its α parameter, so the direction of travel was parallel to the direction of the wind. This can be seen in figure 27 where a wind in the +z direction is applied. For a comparison to the UAV in free-flight,

wind, in the +y direction, was also applied to a UAV using the Feedback Linearization Controller as seen in figure 28.

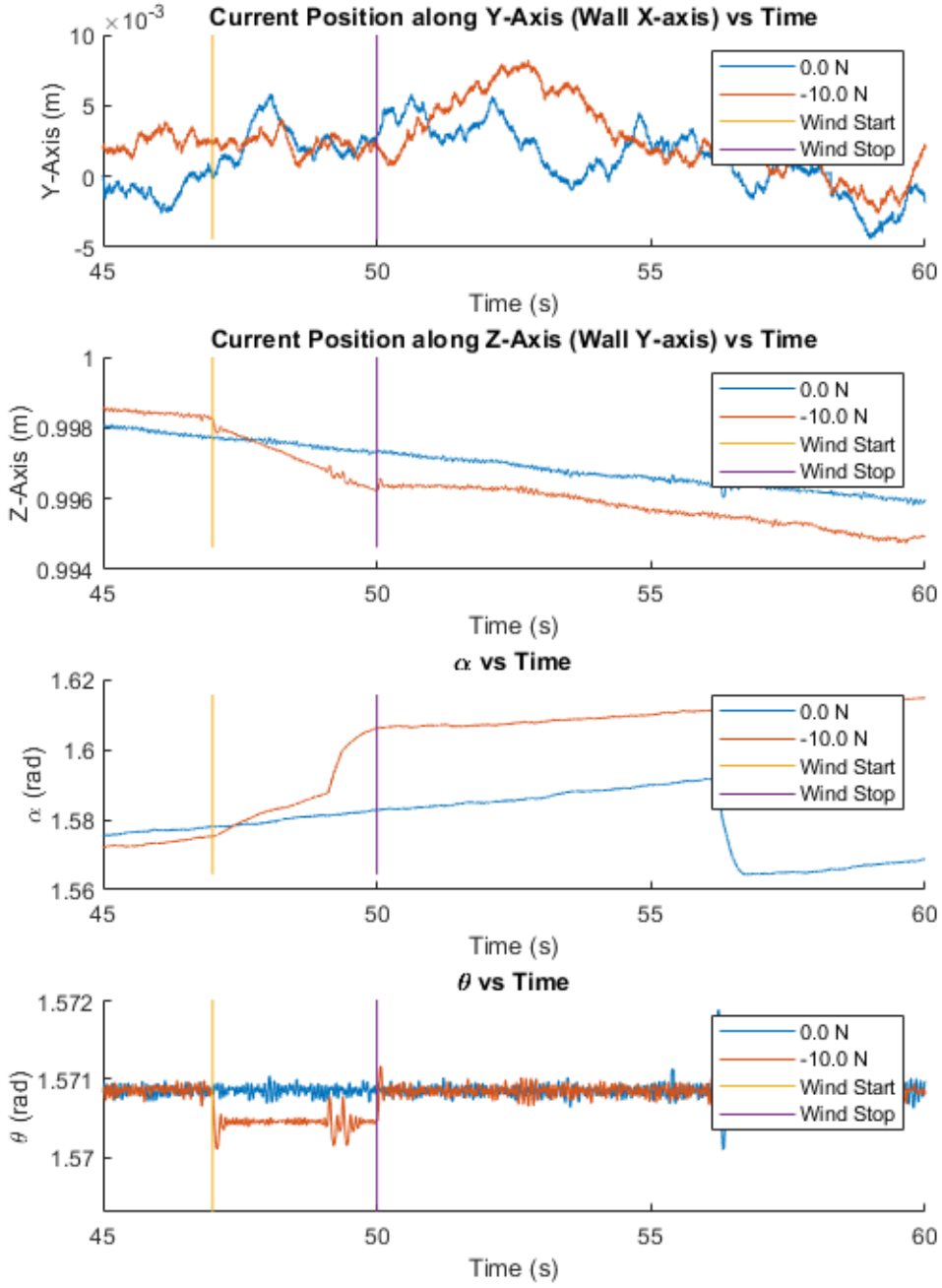

Figure 25: UAV on Wall Resisting Wind Gust from X-Axis

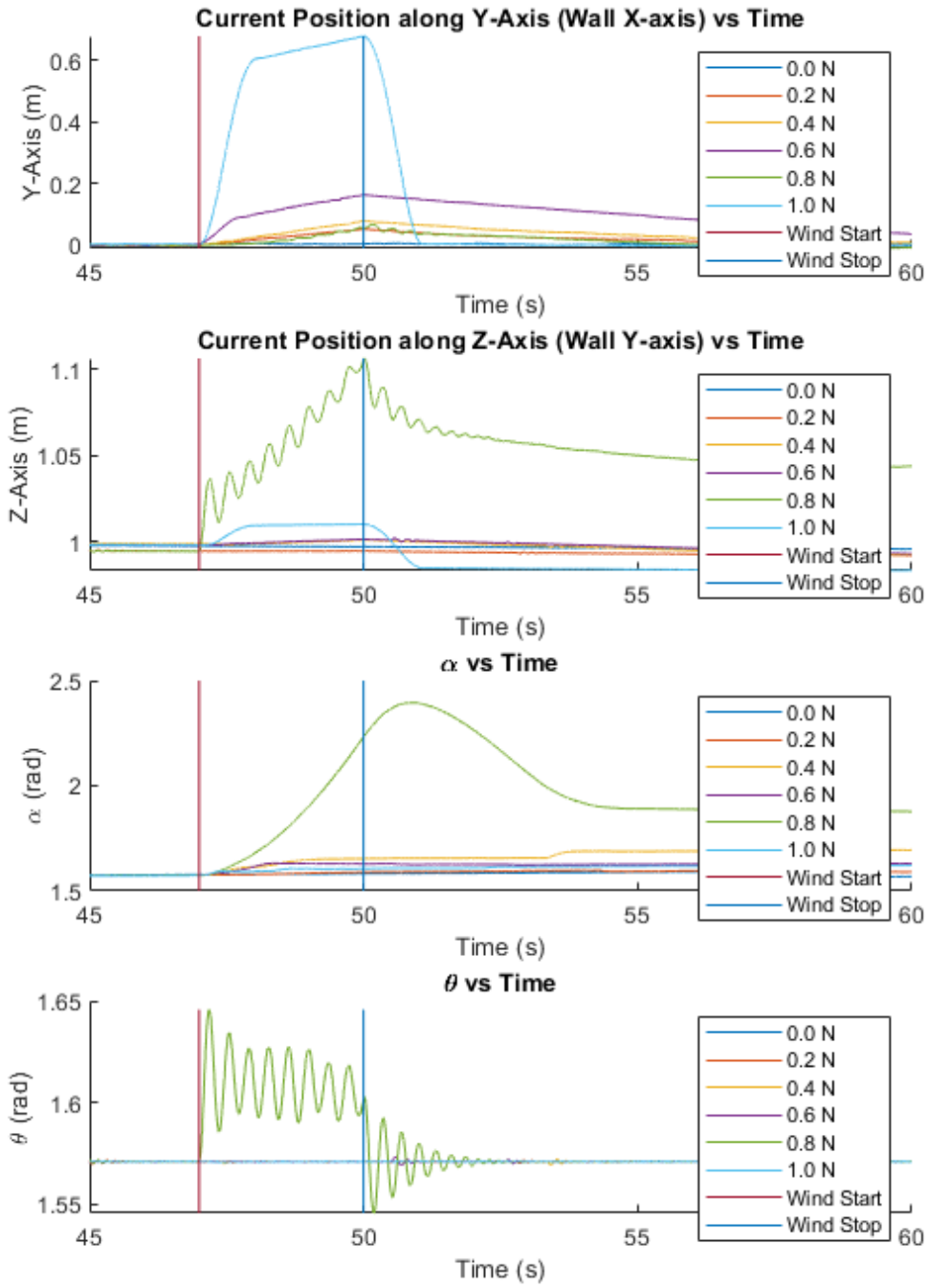

Figure 26: UAV on Wall Resisting Wind Gust from Y-Axis

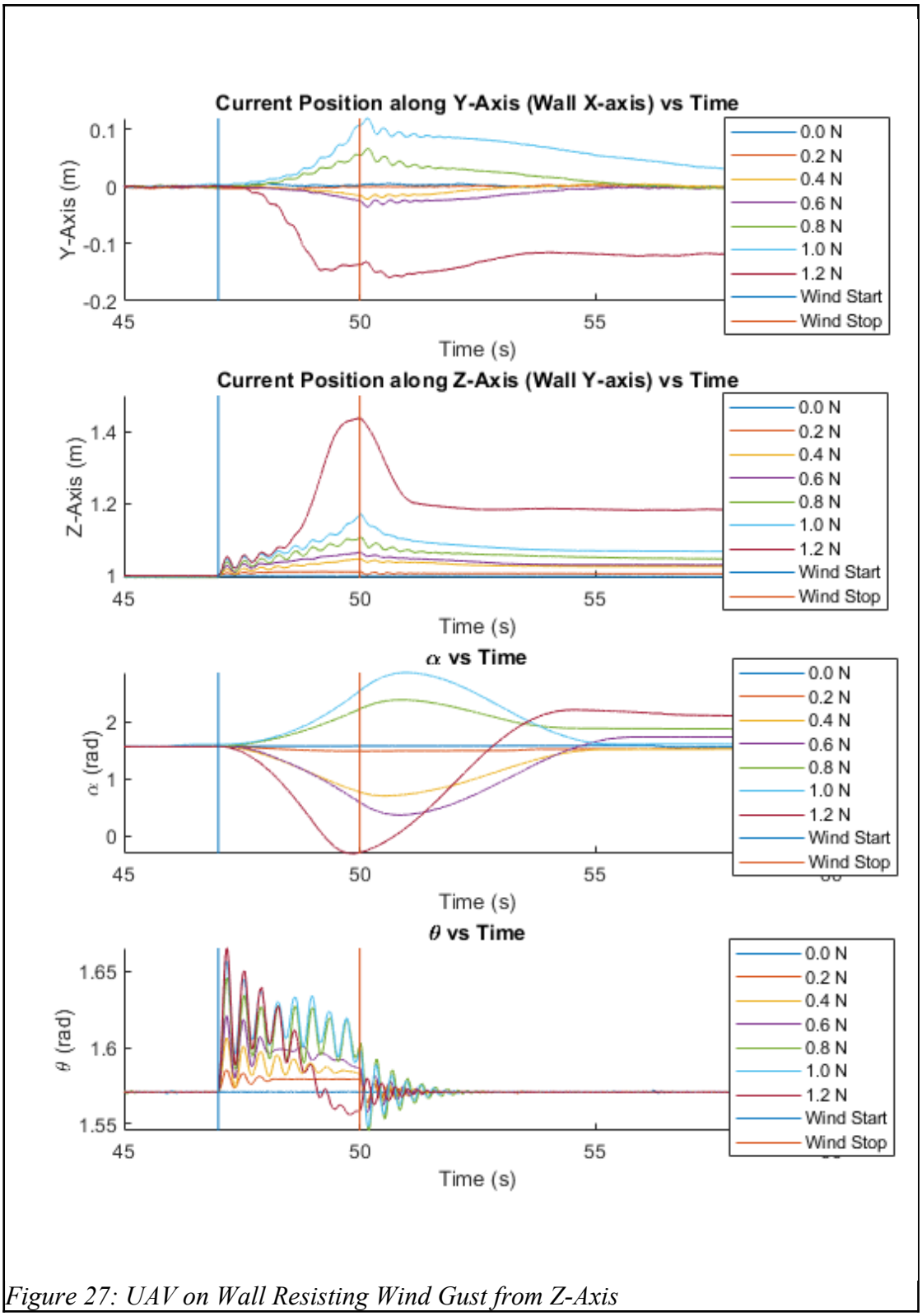

Figure 27: UAV on Wall Resisting Wind Gust from Z-Axis

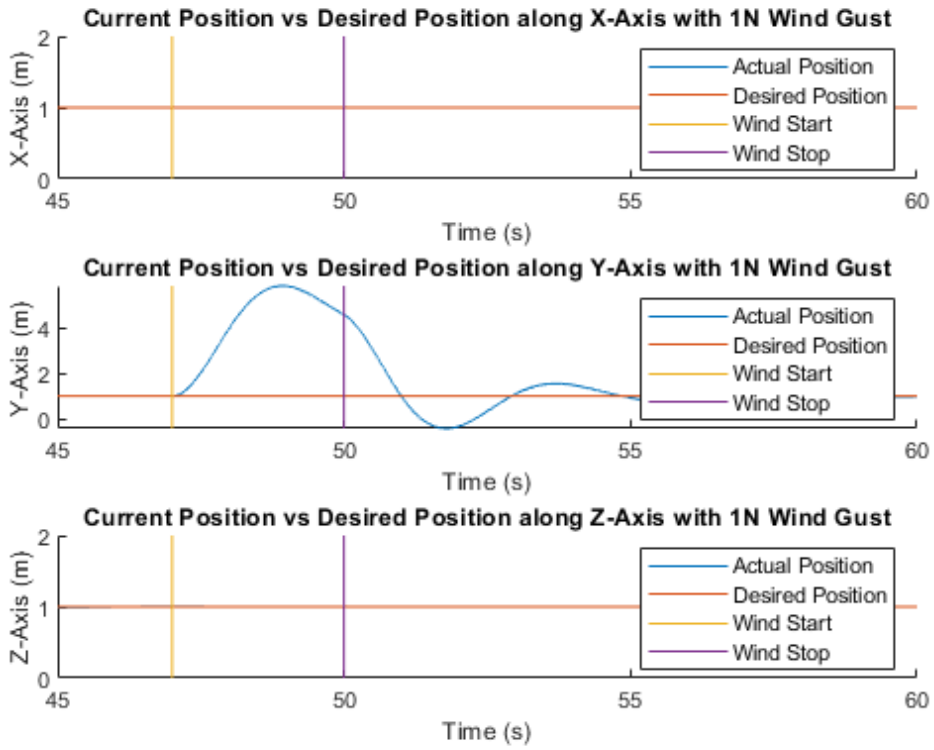

Figure 28: UAV in Free Flight resisting 1N Wing Gust along Y-Axis

CHAPTER 7

CONCLUSION

This thesis presents a novel solution for a UAV to travel through an environment. By attaching a rigid ring around the UAV and having it roll along a surface, the UAV is able to move closer to a target point while using the wall as an anchor. The movement required to create the desired rolling motion was calculated using a series of 3 Denavit-Hartenberg frames accompanied by a velocity of the base frame. By giving the Kinematic Rolling Model parameters a trajectory the model can be used to generate a trajectory in space. Plugging this trajectory in space into an altered Dynamic equation provided the required forces and torques and as well as the required motor speed. This wall rolling method was able to move a simulated UAV across the wall.

The UAV did experience some limitations in its movement due to needing to keep the stability requirements of the UAV and the nonholonomic constraints of using rolling motion. The wall rolling motion wall also able to be integrated into a UAV by including the control methods for the wall rolling with the free flight controllers under a Nonlinear Model Predictive Controller.

The UAV was able to use the wall as an anchor point to resist wind forces. The wind forces was able to move the UAV, but is was significantly less then if the UAV was in free-flight. The UAV was also able to transfer the forces through the body of the UAV into the wall instead of gaining momentum and crashing into the wall.

REFERENCES

[1] Wang BH, Wang DB, Ali ZA, Ting Ting B, Wang H. An overview of various kinds of wind effects on unmanned aerial vehicle. *Measurement and Control*. 2019;52(7-8):731-739. doi:10.1177/0020294019847688

[2] Gabriele Perozzi, Denis Efimov, Jean-Marc Biannic, Laurent Planckaert, Patricia Coton. Wind rejection via quasi-continuous sliding mode technique to control safely a mini drone. *7th European Conference for Aeronautics and Space Science*, Jul 2017, Milan, Italy. 〈hal-01566857〉

[3] S. Mizutani, Y. Okada, C. J. Salaan, T. Ishii, K. Ohno and S. Tadokoro, "Proposal and experimental validation of a design strategy for a UAV with a passive rotating spherical shell," *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Hamburg, 2015, pp. 1271-1278, doi: 10.1109/IROS.2015.7353532.

[4] Andeweg, Sem. "Unmanned aerial vehicle for positioning against a wall." U.S. Patent Application No. 16/644,983.

[5] Mattar, R.A.; Kalai, R. Development of a Wall-Sticking Drone for Non-Destructive Ultrasonic and Corrosion Testing. Drones 2018, 2, 8.

[6] G. Gioioso, M. Ryll, D. Prattichizzo, H. H. Bülthoff and A. Franchi, "Turning a near-hovering controlled quadrotor into a 3D force effector," 2014 IEEE International Conference on Robotics and Automation (ICRA), Hong Kong, 2014, pp. 6278-6284, doi: 10.1109/ICRA.2014.6907785.

[7] S. Rajappa, H. H. Bülthoff, M. Odelga and P. Stegagno, "A control architecture for physical human-UAV interaction with a fully actuated

hexarotor," 2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Vancouver, BC, 2017, pp. 4618-4625, doi: 10.1109/IROS.2017.8206332.

[8] S. Rajappa, M. Ryll, H. H. Bülthoff and A. Franchi, "Modeling, control and design optimization for a fully-actuated hexarotor aerial vehicle with tilted propellers," 2015 IEEE International Conference on Robotics and Automation (ICRA), Seattle, WA, 2015, pp. 4006-4013, doi: 10.1109/ICRA.2015.7139759.

[9] Bicego, D., Mazzetto, J., Carli, R. et al. Nonlinear Model Predictive Control with Enhanced Actuator Model for Multi-Rotor Aerial Vehicles with Generic Designs. J Intell Robot Syst 100, 1213–1247 (2020). https://doi.org/10.1007/s10846-020-01250-9

[10] M. Odelga, P. Stegagno and H. H. Bülthoff, "A fully actuated quadrotor UAV with a propeller tilting mechanism: Modeling and control," 2016 IEEE International Conference on Advanced Intelligent Mechatronics (AIM), Banff, AB, 2016, pp. 306-311, doi: 10.1109/AIM.2016.7576784.

[11] Grüne L., Pannek J. (2011) Nonlinear Model Predictive Control. In: Nonlinear Model Predictive Control. Communications and Control Engineering. Springer, London. https://doi.org/10.1007/978-0-85729-501-9_3

[12] https://www.dji.com/, last accessed on 12/29/2020

[13] https://www.parrot.com/us/drones, last accessed on 12/29/2020

[14] Convens, Bryan, Kelly Merckaert, Marco M. Nicotra, Roberto Naldi, and Emanuele Garone. "Control of fully actuated unmanned aerial vehicles with actuator saturation." *IFAC-PapersOnLine* 50, no. 1 (2017): 12715-12720.

[15] K. Konolige, "A gradient method for realtime robot control," *Proceedings. 2000 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2000) (Cat. No.00CH37113)*, 2000, pp. 639-646 vol.1, doi: 10.1109/IROS.2000.894676.

[16] R. A. Suárez Fernández, S. Dominguez and P. Campoy, "L1 adaptive control for Wind gust rejection in quad-rotor UAV wind turbine inspection," *2017 International Conference on Unmanned Aircraft Systems (ICUAS)*, 2017, pp. 1840-1849, doi: 10.1109/ICUAS.2017.7991485.

BIBLIOGRAPHY

Andeweg, Sem. "Unmanned aerial vehicle for positioning against a wall." U.S. Patent Application No. 16/644,983.

Bicego, D., Mazzetto, J., Carli, R. et al. Nonlinear Model Predictive Control with Enhanced Actuator Model for Multi-Rotor Aerial Vehicles with Generic Designs. J Intell Robot Syst 100, 1213–1247 (2020). https://doi.org/10.1007/s10846-020-01250-9

Convens, Bryan, Kelly Merckaert, Marco M. Nicotra, Roberto Naldi, and Emanuele Garone. "Control of fully actuated unmanned aerial vehicles with actuator saturation." *IFAC-PapersOnLine* 50, no. 1 (2017): 12715-12720.

G. Gioioso, M. Ryll, D. Prattichizzo, H. H. Bülthoff and A. Franchi, "Turning a near-hovering controlled quadrotor into a 3D force effector," 2014 IEEE International Conference on Robotics and Automation (ICRA), Hong Kong, 2014, pp. 6278-6284, doi: 10.1109/ICRA.2014.6907785.

Grüne L., Pannek J. (2011) Nonlinear Model Predictive Control. In: Nonlinear Model Predictive Control. Communications and Control Engineering. Springer, London. https://doi.org/10.1007/978-0-85729-501-9_3

K. Konolige, "A gradient method for realtime robot control," *Proceedings. 2000 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2000) (Cat. No.00CH37113)*, 2000, pp. 639-646 vol.1, doi: 10.1109/IROS.2000.894676.

Mattar, R.A.; Kalai, R. Development of a Wall-Sticking Drone for Non-Destructive Ultrasonic and Corrosion Testing. Drones 2018, 2, 8.

S. Mizutani, Y. Okada, C. J. Salaan, T. Ishii, K. Ohno and S. Tadokoro, "Proposal and experimental validation of a design strategy for a UAV with a passive rotating spherical shell," *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Hamburg, 2015, pp. 1271-1278, doi: 10.1109/IROS.2015.7353532.

M. Odelga, P. Stegagno and H. H. Bülthoff, "A fully actuated quadrotor UAV with a propeller tilting mechanism: Modeling and control," 2016 IEEE International Conference on Advanced Intelligent Mechatronics (AIM), Banff, AB, 2016, pp. 306-311, doi: 10.1109/AIM.2016.7576784.

Gabriele Perozzi, Denis Efimov, Jean-Marc Biannic, Laurent Planckaert, Patricia Coton. Wind rejection via quasi-continuous sliding mode technique to control safely a mini drone. *7th European Conference for Aeronautics and Space Science*, Jul 2017, Milan, Italy. 〈hal-01566857〉

S. Rajappa, H. H. Bülthoff, M. Odelga and P. Stegagno, "A control architecture for physical human-UAV interaction with a fully actuated hexarotor," 2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Vancouver, BC, 2017, pp. 4618-4625, doi: 10.1109/IROS.2017.8206332.

S. Rajappa, M. Ryll, H. H. Bülthoff and A. Franchi, "Modeling, control and design optimization for a fully-actuated hexarotor aerial vehicle with tilted propellers," 2015 IEEE International Conference on Robotics and

Automation (ICRA), Seattle, WA, 2015, pp. 4006-4013, doi: 10.1109/ICRA.2015.7139759.

R. A. Suárez Fernández, S. Dominguez and P. Campoy, "L1 adaptive control for Wind gust rejection in quad-rotor UAV wind turbine inspection," *2017 International Conference on Unmanned Aircraft Systems (ICUAS)*, 2017, pp. 1840-1849, doi: 10.1109/ICUAS.2017.7991485.

Wang BH, Wang DB, Ali ZA, Ting Ting B, Wang H. An overview of various kinds of wind effects on unmanned aerial vehicle. *Measurement and Control*. 2019;52(7-8):731-739. doi:10.1177/0020294019847688

https://www.dji.com/, last accessed on 12/29/2020

https://www.parrot.com/us/drones, last accessed on 12/29/2020